VIETNAM NATIONAL UNIVERSITY, HO CHI MINH CITY
UNIVERSITY OF TECHNOLOGY
FACULTY OF COMPUTER SCIENCE AND ENGINEERING

DATABASE SYSTEMS LABS (CO2014)

Assignment 2

Designing and Implementing Physical Database

Advisor: Dr. Phan Trọng Nhân
Class: Database Systems (Lab) - CC03
Students: Trần Quốc Hoàn - 1952051
Trần Quốc Việt - 1953097
Trần Anh Dũng - 1852306
Ủ Minh Quân - 1911940

HO CHI MINH CITY, NOVEMBER 2021

Contents

1 Member list & Workload

No.	Fullname	Student ID	Problems	Workload
1	Trần Quốc Hoàn	1952051	- Data Insertion - SQL: Procedure (d) - App: Part 3	24%
2	Trần Quốc Việt	1953097	- Data Explanation - SQL: Update (a) - App: Design + Refine - Report writing	28%
3	Trần Anh Dũng	1852306	- Data Insertion - SQL: Select(b) - App: Part 1	24%
4	Ủ Minh Quân	1911940	- Data Insertion - SQL: Function (c) - App: Part 2	24%

2 Introduction

In this assignment 2, we are going to design and implement the physical database from the previous assignment 1, perform SQL commands on it and also develop an application to manage that database. More specifically, it is the database to store Fabric Agency information.
The platform was used to do this assignment is Oracle SQL Developer using Oracle XE database.
Finally, in **Section 6**, to build the application for database management, we choose the **C# (C Sharp)** programming language.

3 Recall of EERD and Mapping

Recalling the Enhanced Entity-Relationship Diagram and Mapping from EERD to Relational Schema:

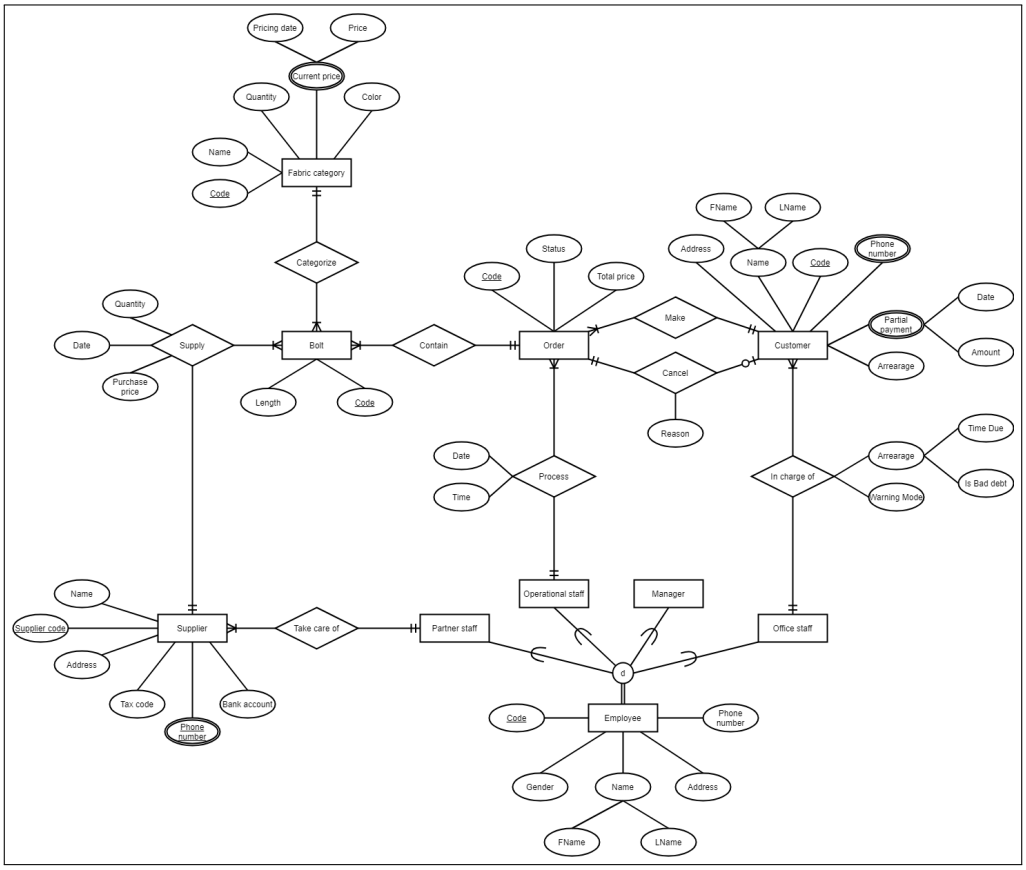


Figure 1: The **Enhanced Entity-Relationship Diagram** designed in Assignment 1

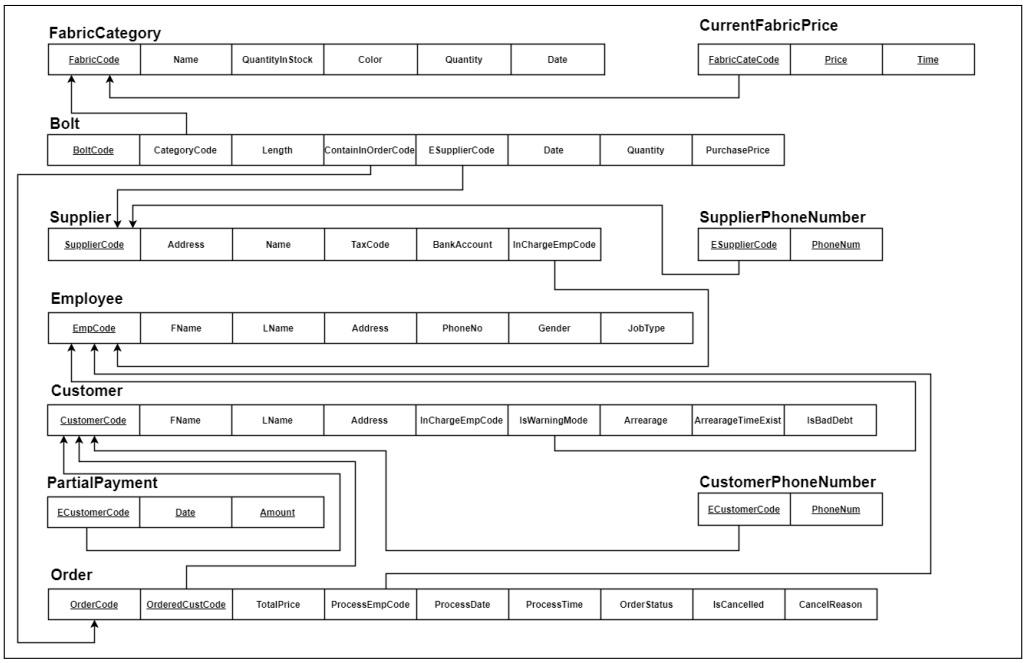


Figure 2: The **Mapping from EERD to Relational Schema** done in Assignment 1

4 Physical Database Design

4.1 Data Insertion

Below code section is the SQL code to drop table (if needed, currently in the comment mode), to create multiple tables, with relationship to other tables (if exist), and to insert data into the database tables.

```sql
DROP TABLE BoltStock CASCADE CONSTRAINTS;
DROP TABLE CurrentPrice CASCADE CONSTRAINTS;
DROP TABLE Customer CASCADE CONSTRAINTS;
DROP TABLE CustomerPhoneNo CASCADE CONSTRAINTS;
DROP TABLE Employee CASCADE CONSTRAINTS;
DROP TABLE OrderList CASCADE CONSTRAINTS;
DROP TABLE Supplier CASCADE CONSTRAINTS;
DROP TABLE FabricCategory CASCADE CONSTRAINTS;
DROP TABLE PartialPayment CASCADE CONSTRAINTS;
DROP TABLE SupplierPhoneNo CASCADE CONSTRAINTS;

CREATE TABLE FabricCategory(
    FabricCode VARCHAR(10) NOT NULL PRIMARY KEY,
    FabricName VARCHAR(20) NOT NULL,
    Quantity INT NOT NULL,
    Color VARCHAR(20)
);

CREATE TABLE CurrentPrice(
    FabricCode VARCHAR(10) NOT NULL,
    Price FLOAT NOT NULL,
```

```sql
    PricedTime DATE NOT NULL,
    PRIMARY KEY (FabricCode, Price, PricedTime),
    CONSTRAINT fk_currentPrice_fabricCode FOREIGN KEY (FabricCode) REFERENCES FabricCategory(FabricCode)
    ON DELETE SET NULL DEFERRABLE
);

CREATE TABLE Employee(
    EmployeeCode VARCHAR(10) NOT NULL PRIMARY KEY,
    FName VARCHAR(10) NOT NULL,
    LName VARCHAR(10) NOT NULL,
    Address VARCHAR(50) NOT NULL,
    PhoneNo INT UNIQUE NOT NULL,
    Gender VARCHAR(10) NOT NULL,
    JobType VARCHAR(20) NOT NULL
);

--For supplier, referencing employee
CREATE TABLE Supplier(
    SupplierCode VARCHAR(10) NOT NULL PRIMARY KEY,
    Address VARCHAR(50) NOT NULL,
    Name VARCHAR(20) NOT NULL,
    TaxCode VARCHAR(20) NOT NULL UNIQUE,
    BankAccount VARCHAR(30) NOT NULL UNIQUE,
    SuperEmployeeCode VARCHAR(10) NOT NULL,
    CONSTRAINT fk_supplier_superEmployeeCode FOREIGN KEY (SuperEmployeeCode) REFERENCES Employee(EmployeeCode)
    ON DELETE SET NULL DEFERRABLE
);

CREATE TABLE SupplierPhoneNo(
    ESupplierCode VARCHAR(10) NOT NULL,
    PhoneNo INT NOT NULL UNIQUE,
    PRIMARY KEY (ESupplierCode, PhoneNo),
    CONSTRAINT fk_supplierPhone_supplierCode FOREIGN KEY (ESupplierCode) REFERENCES Supplier(SupplierCode)
    ON DELETE SET NULL DEFERRABLE
);

-- For customer, referencing employee
CREATE TABLE Customer(
    CustomerCode VARCHAR(10) NOT NULL PRIMARY KEY,
    FName VARCHAR(10) NOT NULL,
    LName VARCHAR(10) NOT NULL,
    Address VARCHAR(50) NOT NULL,
    SuperEmployeeCode VARCHAR(10) NOT NULL,
    IsWarningMode VARCHAR(10) NOT NULL,
    Arrearage FLOAT NOT NULL,
    ArrearageTimeExist INT NOT NULL,
    IsBadDebt VARCHAR(10) NOT NULL,
    CONSTRAINT fk_customer_superEmployeeCode FOREIGN KEY (SuperEmployeeCode) REFERENCES Employee(EmployeeCode)
    ON DELETE SET NULL DEFERRABLE
);

CREATE TABLE CustomerPhoneNo(
    ECustomerCode VARCHAR(10) NOT NULL,
    PhoneNo INT NOT NULL UNIQUE,
    PRIMARY KEY (ECustomerCode, PhoneNo),
    CONSTRAINT fk_customerPhone_customerCode FOREIGN KEY (ECustomerCode) REFERENCES Customer(CustomerCode)
    ON DELETE SET NULL DEFERRABLE
```

```sql
);

CREATE TABLE PartialPayment(
    ECustomerCode VARCHAR(10) NOT NULL,
    PaymentDate DATE NOT NULL ,
    Amount FLOAT NOT NULL,
    PRIMARY KEY (ECustomerCode, PaymentDate, Amount),
    CONSTRAINT fk_parPayment_customerCode FOREIGN KEY (ECustomerCode) REFERENCES Customer(CustomerCode)
    ON DELETE SET NULL DEFERRABLE
);

-- For Orderlist, referencing customer and employee
CREATE TABLE OrderList(
    OrderCode VARCHAR(10) NOT NULL PRIMARY KEY,
    OrderCustomerCode VARCHAR(10) NOT NULL,
    TotalPrice FLOAT NOT NULL,
    ProcessEmployeeCode VARCHAR(10) NOT NULL,
    ProcessDate DATE NOT NULL,
    OrderStatus VARCHAR(20) NOT NULL,
    IsCancelled VARCHAR(10),
    CancelReason VARCHAR(50),
    CONSTRAINT fk_order_customerCode FOREIGN KEY (OrderCustomerCode) REFERENCES Customer(CustomerCode)
    ON DELETE SET NULL DEFERRABLE,
    CONSTRAINT fk_order_empCode FOREIGN KEY (ProcessEmployeeCode) REFERENCES Employee(EmployeeCode)
    ON DELETE SET NULL DEFERRABLE
);

--For boltstock, referencing fabriccategory and orderlist
CREATE TABLE BoltStock(
    BoltCode VARCHAR(10) NOT NULL PRIMARY KEY,
    CategoryCode VARCHAR(10) NOT NULL,
    Length int NOT NULL,
    ContainInOrderCode VARCHAR(10),
    ESupplierCode VARCHAR(10) NOT NULL,
    DateImported DATE NOT NULL,
    Quantity int NOT NULL,
    PurchasePrice FLOAT NOT NULL,
    CONSTRAINT fk_boltStock_categoryCode FOREIGN KEY (CategoryCode) REFERENCES FabricCategory(FabricCode)
    ON DELETE SET NULL DEFERRABLE,
    CONSTRAINT fk_boltStock_containOrderCode FOREIGN KEY (ContainInOrderCode) REFERENCES OrderList(OrderCode)
    ON DELETE SET NULL DEFERRABLE
);

-- Insert data --
ALTER SESSION SET NLS_DATE_FORMAT = 'DD-MM-YYYY';

INSERT INTO EMPLOYEE VALUES ('E001', 'An', 'Vo Hoang', 'HCMC', 0914077077, 'FEMALE', 'Operational Staff');
INSERT INTO EMPLOYEE VALUES ('E002', 'Duc', 'Nguyen Anh', 'HCMC', 0945454212, 'MALE', 'Office Staff');
INSERT INTO EMPLOYEE VALUES ('E003', 'Hoang', 'Tran', 'HCMC', 0973123391, 'MALE', 'Office Staff');
INSERT INTO EMPLOYEE VALUES ('E004', 'Khang', 'Nham Hoang', 'HCMC', 0962911119, 'MALE', 'Manager');
INSERT INTO EMPLOYEE VALUES ('E005', 'Thanh', 'Le Nhat', 'HCMC', 0936129765, 'FEMALE', 'Partner Staff');

```

```sql
INSERT INTO CUSTOMER VALUES ('CC001', 'Binh', 'Le Anh', 'HCMC', 'E004', 'NO', 1000, 0, 'NO');
INSERT INTO CUSTOMER VALUES ('CC002', 'Dung', 'Huynh Anh', 'Da Nang', 'E002', 'NO', 0, 0, 'NO');
INSERT INTO CUSTOMER VALUES ('CC003', 'Hang', 'Phan Thi', 'Ha Noi', 'E001', 'YES', 1250, 181, 'NO');
INSERT INTO CUSTOMER VALUES ('CC004', 'Linh', 'Nguyen Thi', 'Dong Nai', 'E005', 'NO', 3000, 35, 'YES');
INSERT INTO CUSTOMER VALUES ('CC005', 'Uyen', 'Pham Thanh', 'Long An', 'E003', 'NO', 0, 0, 'NO');

INSERT INTO PARTIALPAYMENT VALUES ('CC001', '13-03-2018', 400);
INSERT INTO PARTIALPAYMENT VALUES ('CC001', '13-09-2018', 1000);
INSERT INTO PARTIALPAYMENT VALUES ('CC002', '30-03-2019', 500.45);
INSERT INTO PARTIALPAYMENT VALUES ('CC003', '28-01-2020', 150);
INSERT INTO PARTIALPAYMENT VALUES ('CC005', '19-04-2021', 4000);

INSERT INTO CUSTOMERPHONENO VALUES ('CC001', 0915055055);
INSERT INTO CUSTOMERPHONENO VALUES ('CC002', 0982177277);
INSERT INTO CUSTOMERPHONENO VALUES ('CC003', 0127488889);
INSERT INTO CUSTOMERPHONENO VALUES ('CC004', 0813921921);
INSERT INTO CUSTOMERPHONENO VALUES ('CC005', 0852696969);
INSERT INTO CUSTOMERPHONENO VALUES ('CC003', 0911384351);

INSERT INTO SUPPLIER VALUES ('S001', '501 Hai Ba Trung, HA NOI', 'Cotton Agency', '33250', '131531000001', 'E001');
INSERT INTO SUPPLIER VALUES ('S002', '1028 Hoang Quoc Viet, HA NOI', 'Wool Agency', '13842', '146484000001', 'E002');
INSERT INTO SUPPLIER VALUES ('S003', '65 Hau Giang, HA NOI', 'Demin Agency', '51348', '131654000001', 'E003');
INSERT INTO SUPPLIER VALUES ('S004', '342/14A Ta Quang Buu, HA NOI', 'Velvet Agency', '13510', '134865000003', 'E003');
INSERT INTO SUPPLIER VALUES ('S005', '301 Tran Nguyen Han, DA NANG', 'Silk Agency', '46813', '138946000003', 'E004');
INSERT INTO SUPPLIER VALUES ('S006', '120 Phan Dang Luu, HCMC', 'Misc. Agency', '17426', '766413000004', 'E005');

INSERT INTO SUPPLIERPHONENO VALUES ('S001', '0710441394');
INSERT INTO SUPPLIERPHONENO VALUES ('S001', '0615953918');
INSERT INTO SUPPLIERPHONENO VALUES ('S002', '0162001266');
INSERT INTO SUPPLIERPHONENO VALUES ('S003', '0134856483');
INSERT INTO SUPPLIERPHONENO VALUES ('S003', '0135464683');
INSERT INTO SUPPLIERPHONENO VALUES ('S004', '0646486252');
INSERT INTO SUPPLIERPHONENO VALUES ('S005', '0133468453');
INSERT INTO SUPPLIERPHONENO VALUES ('S005', '0846513153');
INSERT INTO SUPPLIERPHONENO VALUES ('S005', '0456485372');
INSERT INTO SUPPLIERPHONENO VALUES ('S006', '0846551324');

INSERT INTO FABRICCATEGORY VALUES ('FC001', 'Cotton', 1, 'White');
INSERT INTO FABRICCATEGORY VALUES ('FC002', 'Wool', 1, 'Blue');
INSERT INTO FABRICCATEGORY VALUES ('FC003', 'Demin', 2, 'Dark Blue');
INSERT INTO FABRICCATEGORY VALUES ('FC004', 'Velvet', 1, 'Velvet');
INSERT INTO FABRICCATEGORY VALUES ('FC005', 'Silk', 1, 'Silk');

INSERT INTO CURRENTPRICE VALUES ('FC001', 10, '10-02-2019');
INSERT INTO CURRENTPRICE VALUES ('FC002', 30, '11-09-2019');
INSERT INTO CURRENTPRICE VALUES ('FC005', 30, '29-04-2020');
INSERT INTO CURRENTPRICE VALUES ('FC004', 15, '15-12-2020');
INSERT INTO CURRENTPRICE VALUES ('FC003', 50, '01-01-2021');
INSERT INTO CURRENTPRICE VALUES ('FC002', 30, '15-04-2021');
INSERT INTO CURRENTPRICE VALUES ('FC005', 25, '04-06-2021');
INSERT INTO CURRENTPRICE VALUES ('FC005', 40, '04-10-2021');
```

```sql

INSERT INTO ORDERLIST VALUES ('X001', 'CC001', 351.1, 'E001', '21-08-2019', 'FULL PAID', 'NO', ' ');
INSERT INTO ORDERLIST VALUES ('X002', 'CC001', 105, 'E003', '01-09-2020', 'FULL PAID', 'NO',' ' );
INSERT INTO ORDERLIST VALUES ('X003', 'CC002', 165.4, 'E005', '05-09-2020', 'CANCELLED', 'YES', 'Expensive!!');
INSERT INTO ORDERLIST VALUES ('X004', 'CC005', 255, 'E002', '30-03-2021', 'PARTIAL PAID', 'NO', ' ');
INSERT INTO ORDERLIST VALUES ('X005', 'CC003', 300, 'E002', '01-08-2021', 'ORDERED', 'NO', ' ');
INSERT INTO ORDERLIST VALUES ('X006', 'CC004', 400, 'E004', '30-10-2021', 'NEW', 'NO', ' ');

INSERT INTO BOLTSTOCK VALUES ('BC101', 'FC001', 20, 'X001', 'S001', '15-01-2019', 10, 400);
INSERT INTO BOLTSTOCK VALUES ('BC201', 'FC002', 30, 'X002', 'S002', '23-05-2019', 30, 315.1);
INSERT INTO BOLTSTOCK (BoltCode, CategoryCode, Length, ESupplierCode, DateImported, Quantity, PurchasePrice)
    VALUES ('BC301', 'FC003', 20, 'S003', '12-12-2019', 15, 105);
INSERT INTO BOLTSTOCK VALUES ('BC401', 'FC004', 40, 'X003', 'S004', '21-01-2020', 50, 124.5);
INSERT INTO BOLTSTOCK VALUES ('BC501', 'FC005', 30, 'X004', 'S005', '30-06-2020', 100, 77);
INSERT INTO BOLTSTOCK VALUES ('BC302', 'FC003', 20, 'X005', 'S003', '01-02-2021', 25, 249);
INSERT INTO BOLTSTOCK VALUES ('BC303', 'FC003', 30, 'X006', 'S003', '10-10-2021', 31, 300);
```

4.2 Data explanation

- For the identifying code (Fabric category code, customer code,...etc), we use the data type VARCHAR with the length is 10.
- For the name (Customer name, agency name,... etc), we also use the VARCHAR with longer length is 20.
- For some other information such as Fabric's color, Job type, Supplier's bank account, customer's address,..., we use the data type VARCHAR with varied length (20, 30, 50).
- INTEGER data type is used to define the discrete values like fabric's quantity in stock, Phone number, existed days of arrearage,..
- The fabric price, payment amount and customer's arrearage is listed in FLOAT.
- Data that must follow the Not null-able, Primary key or Foreign key constraints are listed after the data declaration in the SQL initialization code as NOT NULL, PRIMARY KEY or FOREIGN KEY.

4.3 Data testing

This part is to check whether the data has been inserted properly by some basic SELECT commands:

```sql
SELECT * FROM BoltStock;

SELECT LName, FName FROM EMPLOYEE;
```

The result appears as follow, indicating that our insertion has been processed successfully.

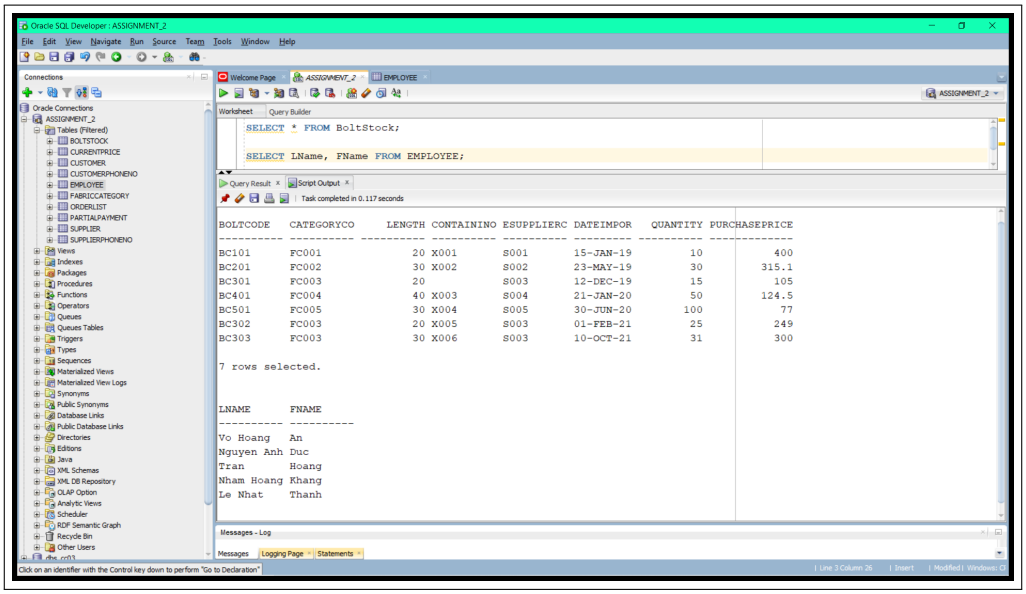


Figure 3: The result of some basic SELECT commands

5 Store Procedure / Function / SQL

5.1 SQL

5.1.1 UPDATE query

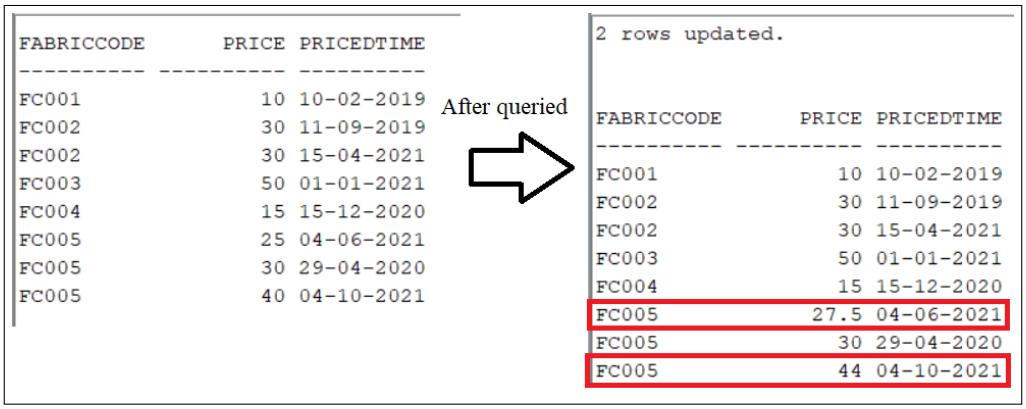
This is the first question and also the code snippet to solve it:
- *Increase Silk selling price to 10% of those provided by all suppliers from 01/09/2020.*

```sql
SELECT * FROM CurrentPrice;

UPDATE CurrentPrice
SET Price = Price*1.1
WHERE CurrentPrice.FabricCode IN
(
    SELECT FabricCategory.FabricCode
    FROM FabricCategory
    WHERE FabricCategory.FabricName = 'Silk'
)
AND CurrentPrice.PricedTime > '01-Sep-2020'
AND CurrentPrice.PricedTime IS NOT NULL;

SELECT * FROM CurrentPrice;
```

SQL Developer will run the query and return like this, comparing to the original data before modified, using the command SELECT * FROM CurrentPrice;:



FABRICCODE	PRICE	PRICEDTIME
FC001	10	10-02-2019
FC002	30	11-09-2019
FC002	30	15-04-2021
FC003	50	01-01-2021
FC004	15	15-12-2020
FC005	25	04-06-2021
FC005	30	29-04-2020
FC005	40	04-10-2021

2 rows updated.

FABRICCODE	PRICE	PRICEDTIME
FC001	10	10-02-2019
FC002	30	11-09-2019
FC002	30	15-04-2021
FC003	50	01-01-2021
FC004	15	15-12-2020
FC005	27.5	04-06-2021
FC005	30	29-04-2020
FC005	44	04-10-2021

Figure 4: The data after UPDATE command

5.1.2 SELECT query

For the second question, we will perform some SELECT queries:
- *Select all orders containing bolt from the supplier named 'Silk Agency'.*

```
SELECT OrderList.OrderCode AS ORDER_CODE,
       OrderList.OrderCustomerCode AS CUSTOMER_CODE,
       OrderList.TotalPrice AS TOTAL_PRICE,
       OrderList.ProcessEmployeeCode AS PROCESS_BY,
       OrderList.ProcessDate AS PROCESS_DATE,
       OrderList.IsCancelled AS IS_CANCELLED,
       OrderList.CancelReason AS CANCEL_REASON
FROM BoltStock, OrderList, Supplier
WHERE
(
    BoltStock.ContainInOrderCode = OrderList.OrderCode
    AND BoltStock.ESupplierCode = Supplier.SupplierCode
)
AND Supplier.Name = 'Silk Agency';
```

The SELECT query returns the table as:

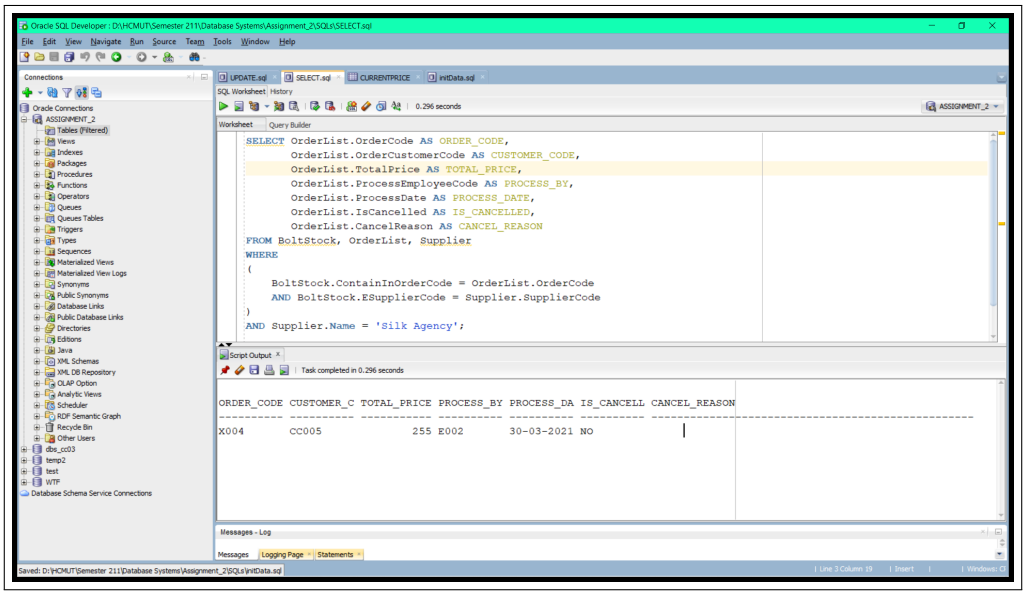


Figure 5: The returned table after SELECT command

5.2 Function

Thirdly, it is a very interesting paradigm of any programming languages, the FUNCTION. We need to do the following task:
- *Write a function to calculate the total purchase price the agency has to pay for each supplier.*
Below is the code snippet to create a function and to test it:

```
CREATE OR REPLACE FUNCTION supplier_payment(supplier_id in VARCHAR2)
RETURN DECIMAL
    AS TotalPrice DECIMAL(19, 4);
BEGIN
    SELECT
        SUM(B.PurchasePrice) INTO TotalPrice
    FROM
        Supplier S, BoltStock B
    WHERE
        S.SupplierCode = supplier_id AND
        S.SupplierCode = B.ESupplierCode;
    --GROUP BY S.SupplierCode;
    RETURN TotalPrice;
END;
```

To test the function:

```
SELECT supplier_payment('S003') AS TOTAL_PRICE FROM DUAL;
```

The total result will be calculated and displayed as follow when we calculate the total purchased price for Supplier S003:

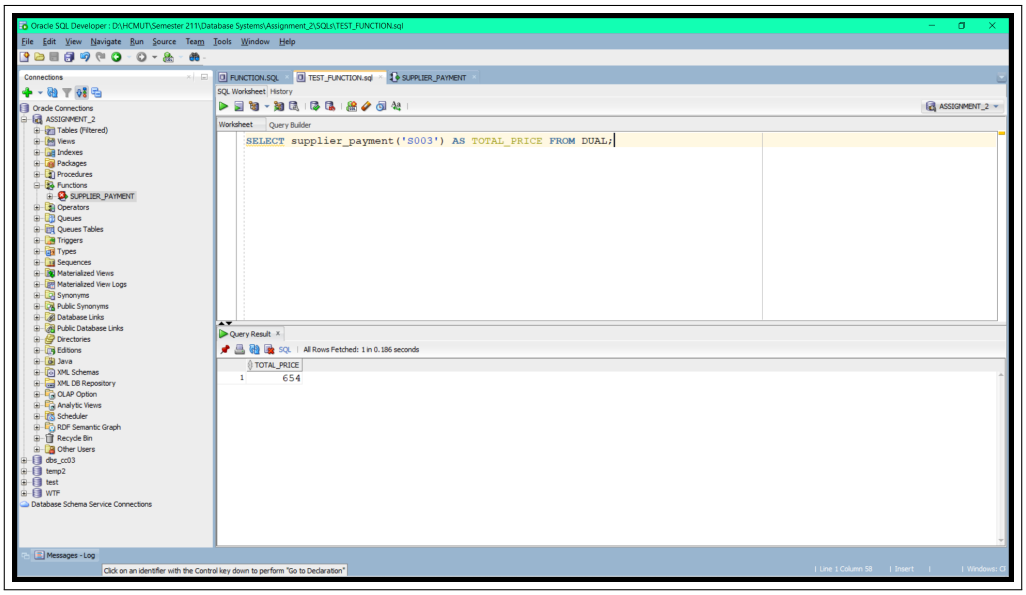


Figure 6: The result of FUNCTION command

5.3 Stored Procedure

Finally, we will implement the PROCEDURE command in SQL to deal with this problem:
- *Write a procedure to sort the suppliers in increasing number of categories they provide in a period of time.*
Below is the SQL code snippet to create a procedure along with one unit test:

```
--DROP PROCEDURE sort_Supplier;
CREATE OR REPLACE PROCEDURE sort_Supplier(
    lower_date IN DATE,
    upper_date IN DATE
)
AS
BEGIN
    FOR supplier_record IN(
        -- sort supplier in increasing number of categories provide
        SELECT suppliercode AS Scode, COUNT(Esuppliercode) AS count_supply
        FROM (
            SELECT DISTINCT S.suppliercode, B.Esuppliercode
            FROM supplier S LEFT JOIN boltstock B
            ON S.suppliercode = B.esuppliercode
                AND CAST(dateimported AS DATE) < CAST(upper_date AS DATE)
                AND CAST(dateimported AS DATE) > CAST(lower_date AS DATE)
        )
        GROUP BY suppliercode
        ORDER BY count_supply
    )
    -- print out to console
    LOOP
        dbms_output.put_line('Supplier_id: ' || supplier_record.Scode
                             || ' provide : ' || supplier_record.count_supply || '
    categories');
    END LOOP;
```

```
END;
```

To test the stored procedure

```
EXEC sort_Supplier('10-Dec-2019', '10-Dec-2021');
```

The result would appear as:

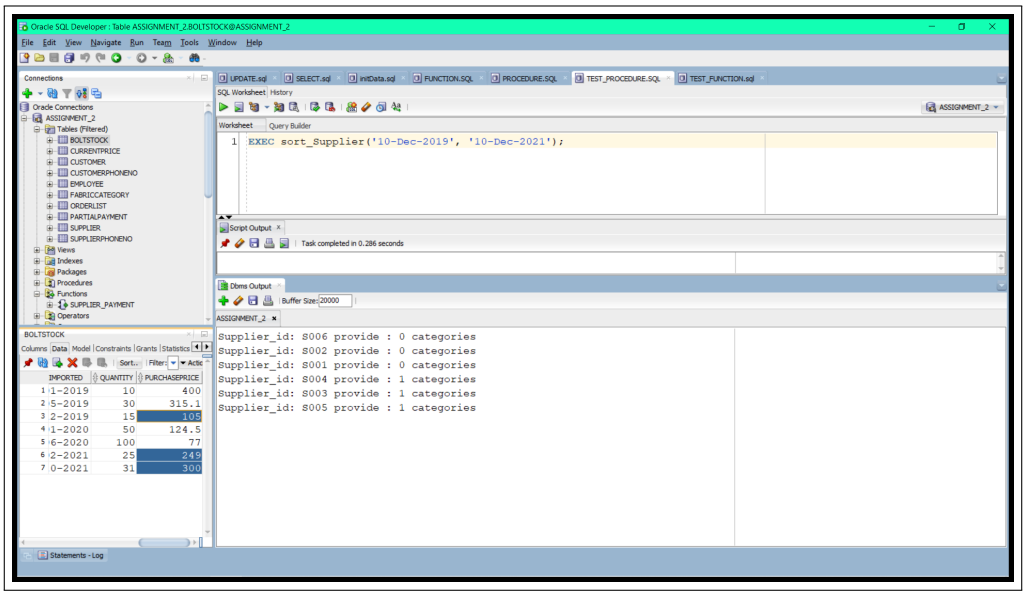


Figure 7: The DBMS output of PROCEDURE

6 Building Application

6.1 Prerequisite

Before diving into the application, we must perform some steps using SQL developer to create and to grant proper privileges to an administrative user. Connect to or create a connection using the SYSTEM user and execute the following SQL commands:

```sql
-- Create the user that will be used
ALTER SESSION SET "_oracle_script" = TRUE;

-- Drop user if already existed
DROP USER Assignment_2 CASCADE;

CREATE USER Assignment_2 IDENTIFIED BY "123456";
GRANT DBA TO Assignment_2;
```

After that, we now have a user with the name is ASSIGNMENT_2, the password is 123456, and is granted most of the DBA privileges. Then we create a new database connection named ASSIGNMENT_2 using the created user above.
Next step, we initialize the data using the SQL code similar to the **Section 4.1 (Data Insertion)**. At this point, all the data should be correctly loaded in.
Finally, we execute this code snippet to create an administrative user with some provided privilege such as SELECT, INSERT, UPDATE, and DELETE.

```sql
-- To create new manager with password = 123456
ALTER SESSION SET "_oracle_script" = TRUE;
DROP USER FABRIC_AGENCY;
CREATE USER FABRIC_AGENCY IDENTIFIED BY "123456";

-- Grant access to the MANAGER
GRANT CREATE SESSION TO FABRIC_AGENCY;
GRANT SELECT, INSERT, UPDATE, DELETE ON Assignment_2.BOLTSTOCK TO FABRIC_AGENCY;
GRANT SELECT, INSERT, UPDATE, DELETE ON Assignment_2.CURRENTPRICE TO FABRIC_AGENCY;
GRANT SELECT, INSERT, UPDATE, DELETE ON Assignment_2.CUSTOMER TO FABRIC_AGENCY;
GRANT SELECT, INSERT, UPDATE, DELETE ON Assignment_2.CUSTOMERPHONENO TO FABRIC_AGENCY;
GRANT SELECT, INSERT, UPDATE, DELETE ON Assignment_2.EMPLOYEE TO FABRIC_AGENCY;
GRANT SELECT, INSERT, UPDATE, DELETE ON Assignment_2.FABRICCATEGORY TO FABRIC_AGENCY;
GRANT SELECT, INSERT, UPDATE, DELETE ON Assignment_2.ORDERLIST TO FABRIC_AGENCY;
GRANT SELECT, INSERT, UPDATE, DELETE ON Assignment_2.PARTIALPAYMENT TO FABRIC_AGENCY;
GRANT SELECT, INSERT, UPDATE, DELETE ON Assignment_2.SUPPLIER TO FABRIC_AGENCY;
GRANT SELECT, INSERT, UPDATE, DELETE ON Assignment_2.SUPPLIERPHONENO TO FABRIC_AGENCY;

GRANT SELECT, INSERT, UPDATE, DELETE ON Assignment_2.ORDER_DETAIL_VIEWS TO FABRIC_AGENCY;

GRANT EXECUTE ON pkg_global_params TO FABRIC_AGENCY;
GRANT EXECUTE ON pkg_customer_params TO FABRIC_AGENCY;
```

We have successfully created a new administrative account to manage the Fabric Agency's database with the following information

```
Username: FABRIC_AGENCY
Password: 123456
```

Let's continue to explore the application in the next sub-section.

6.2 User Manual

6.2.1 Login and Logout

Firstly, start the application with the "lightning bolt" icon. The login UI screen will appear as the *Figure 8* below. Log in to the system using the previously created account, if the authentication step is success, the line **Currently: NOT LOGGED IN** should be changed to **Currently: LOGGED IN AS: FABRIC_AGENCY** like in *Figure 9*. We can also terminate the session by pressing the **Logout** button while logging in.

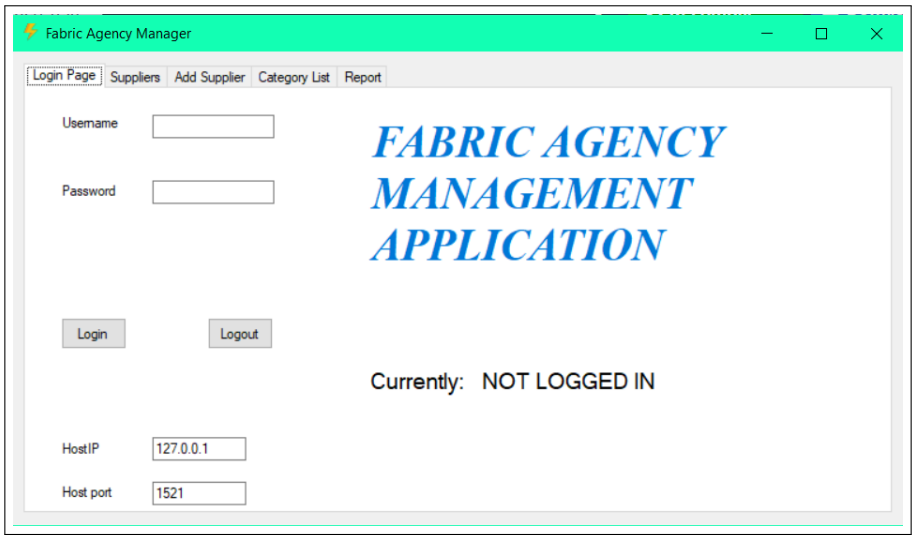


Figure 8: The UI screen of Fabric Agency application

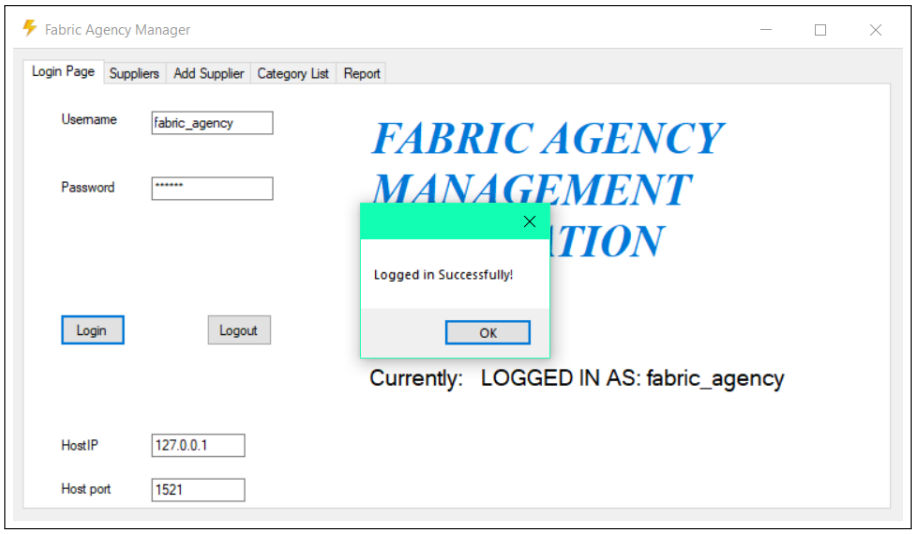


Figure 9: The UI screen after logging in successfully

6.2.2 Search for supplier and supplement

In the second tab, we can either search for supplier's information and supplement records by Supplier ID or by Supplier name. Insert the information corresponding to the search box, and press the **SEARCH** button. If we specify both of the search boxes, SQL will perform an OR operation to display the data. Otherwise, the application will throw an error that we must at least fill in one box.

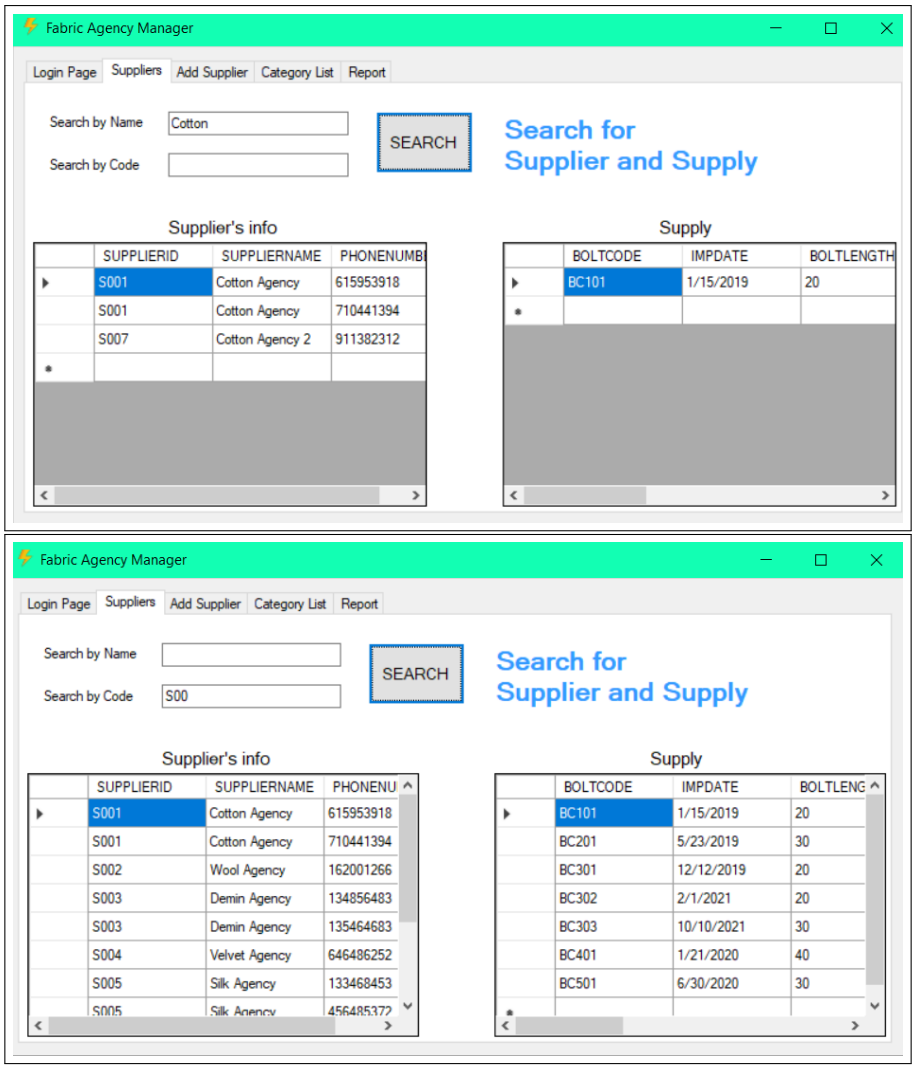


Figure 10: Successfully searched for supplier and corresponding supplement

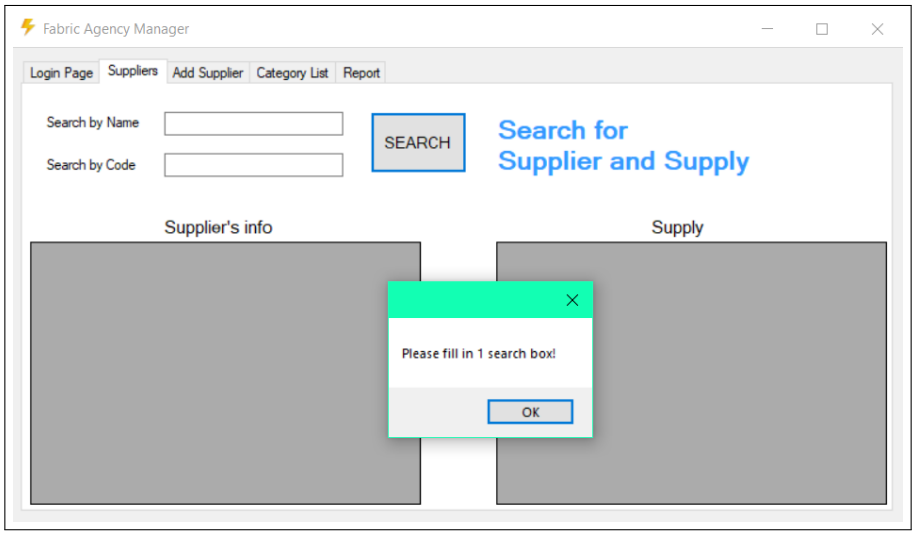


Figure 11: An error thrown by the application if we leave both the search boxes blank

6.2.3 Add a new supplier

In the third tab, we can add a new supplier into the database, by specifying their information such as Name, Address, Tax Code, Bank Account, Phone Number, and Managing Employee. The Supplier ID will be generated automatically based on the current highest supplier ID.
For example: If we have suppliers with the IDs of S001, S002, S003, S004, S005 already in our database, the next supplier to be added will have an ID of S006.
The insertion can only be performed once every blank has been filled and no unique constraint violated.

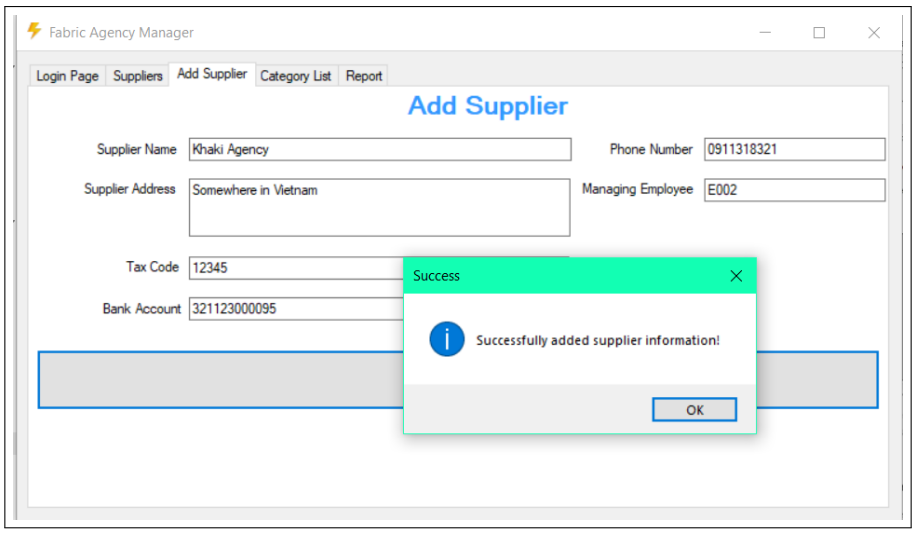


Figure 12: Adding a new supplier successfully

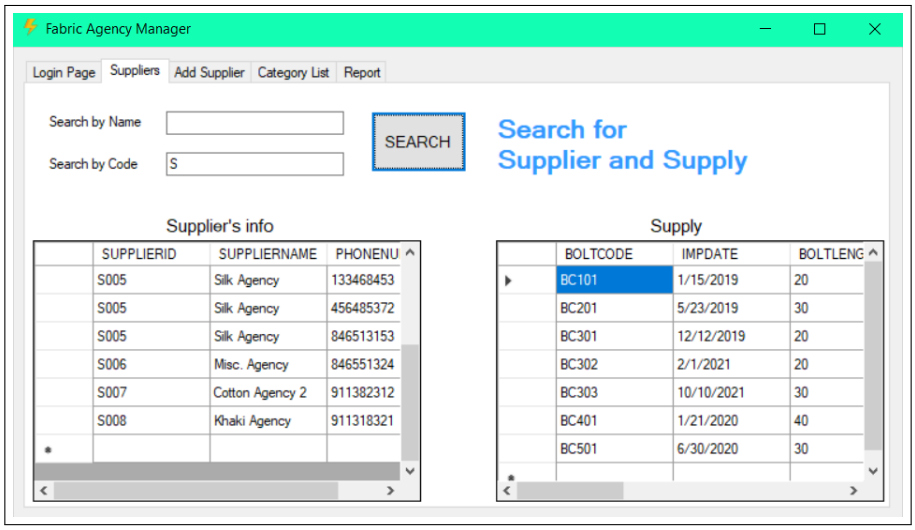


Figure 13: Info of the new supplier added in the database

6.2.4 List category detail

In the fourth tab, we can list out the fabric category information that was provided by a specific supplier using the Supplier ID. There is only one search box to provide the ID, and the result will be displayed if the it was properly found. Otherwise, the application will throw an "Supplier ID not found!" exception.

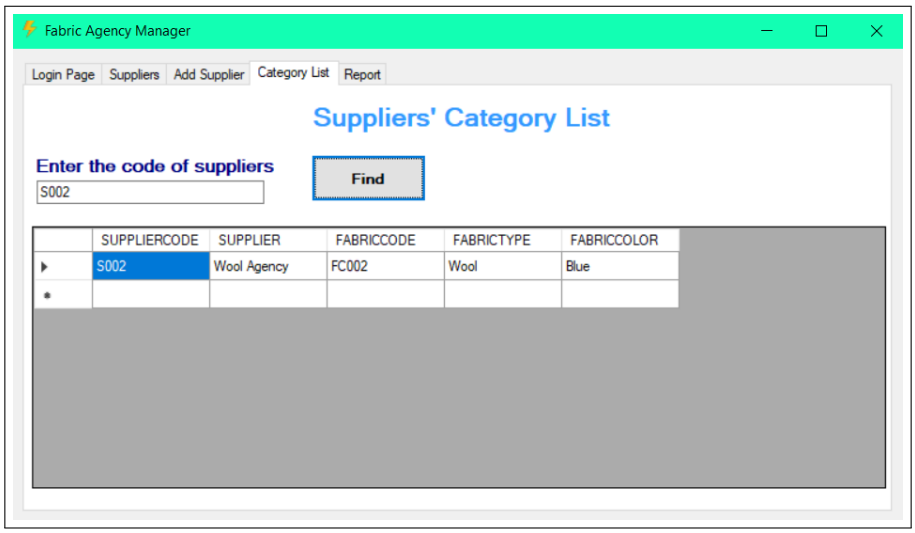


Figure 14: Result of finding the Category information using Supplier ID

6.2.5 Make the report

For this part, we did not implement it.

7 Github source code

For further details, visit our **Github's repository** at this link:
https://github.com/DuckyHCMUT/HK211_DatabaseSystemsLab